Grundlagen der theoretischen Informatik

Vorlesung 3:
Minimale Spannbäume, Breitensuche

Spannbaum mit minimalen Kosten

Definition

Einen Teilgraph B von G heisst einen *Spannbaum (feszítő fa)* falls $V(B) = V(G)$ und B ein Baum ist.

Definition

Sei $G = (V, E)$ ein zusammenhängender Graph und sei $k : E \to \mathbb{R}$ eine Funktion. $k(e)$ gibt die *Kosten (költség)* der Kante e an.

Wenn $F \subseteq E$, dann bezeichne $k(F) = \sum_{e \in F} k(e)$ die Kosten von F.

Ein Spannbaum $B = (V, F)$ von G ist ein *Spannbaum mit minimalen Kosten (minimális költségű feszítő fa)*, wenn $k(F)$ unter allen Spannbäumen von G minimal ist. $k(F)$ wird auch mit $k(B)$ bezeichnet.

Kruskal-Algorithmus oder gieriger (mohó) Algorithmus

Input: G und k wie in Definition
Output: Ein Spannbaum $F_m = F$ von G mit minimalen Kosten.
Ablauf: Sei $E = \{e_1, e_2, e_3, \ldots, e_m\}$ so dass
$k(e_1) \leq k(e_2) \leq \cdots \leq k(e_m)$.
Sei $F_0 = (V, \emptyset)$.
In jedem Schritt wir probieren eine neue Kante aus E dazu nehmen, und zwar eine, die mit den bisher ausgewählten Kanten keinen Kreis bildet.
a) wenn $F_{i-1} \cup \{e_i\}$ kreisfrei ist, sei $F_i = F_{i-1} \cup \{e_i\}$;
b) wenn $F_{i-1} \cup \{e_i\}$ nicht kreisfrei ist, sei $F_i = F_{i-1}$.

Satz

Der Kruskal-Algorithmus liefert einen Spannbaum mit minimalen Kosten.

Beweis. Es ist klar, dass der Algorithmus einen Spannbaum liefert:
1. wir achten darauf, keinen Kreis zu machen;
2. solange die ausgewählten Kanten noch keinen zusammenhängenden Teilgraphen bilden, ist es immer möglich, noch eine weitere Kante auszuwählen.

Sei der vom Algorithmus gelieferte Spannbaum $F = (V, E_1)$. Nehmen wir indirekt an, es gibt einen Spannbaum $B = (V; E_2)$ mit minimalen Kosten, mit $k(B) < k(F)$. Wenn es mehrere gibt, dann nehmen wir einen, bei dem die Anzahl der mit F gemeinsamen Kanten maximal ist ($E_1 \cap E_2$ ist maximal).

Sei $e \in E_2 \setminus E_1$. $E_1 \cup \{e\}$ enthält einen Kreis K. Für eine beliebige Kante e_j in $K - e$ gilt, dass $k(e_j) \leq k(e)$, weil der Algorithmus sonst e statt e_j gewählt hätte.

$B - e$ hat zwei Komponenten. $K - e$ läuft zwischen diesen beiden Komponenten, daher gibt es mindestens eine Kante e_k, die zwischen den beiden Komponenten läuft. Dann muss aber $k(e_k) \geq k(e)$ gelten, sonst wäre $B - e + e_k$ ein Spannbaum mit niedrigeren Kosten als B.
Dann folgt, dass $k(e) = k(e_k)$. Dann ist aber $B - e + e_k$ auch ein Spannbaum mit minimalen Kosten, aber er hat eine Kante mehr gemeinsam mit F als B (nämlich e_k). Das widerspricht aber der Wahl von B.

Anwendung von minimalen Spannbäume in Netzwerkanalyse

Man braucht diese in Signale und Systeme.
Aus den bekannten Werten der Schaltelemente und den vorgegebenen Quellgrössen werden alle Ströhme und Spannungen berechnet werden.

Die grosse Frage:
Wenn ist die Lösung eindeutig?

Definition (Schnitt)

Eine Kantenmenge $Q \subseteq E(G)$ ist ein *Schnitt* falls $G - Q$ mehrere Komponente als G hat aber Q hat keine Teilmenge mit dieser Eigenschaft.

Der Knotenpunktsatz (Knotenregel) – 1. Kirchhoffsches Gesetz

In einem Knotenpunkt eines elektrischen Netzwerkes ist die Summe der zufließenden Ströme gleich der Summe der abfließenden Ströme.

$\Rightarrow$ Es gibt keinen Schnitt der nur aus Stromquellen besteht.

Der Maschensatz (Maschenregel) – 2. Kirchhoffsches Gesetz

Alle Teilspannungen eines Umlaufs (bzw. einer Masche) in einem elektrischen Netzwerk addieren sich zu null.

$\Rightarrow$ Es gibt keinen Kreis der nur aus Spannungsquellen besteht.

Definition (Normalbaum)

Gegeben ist ein zusammenhängender Graph dessen Kanten Spannungsquellen, Stromquellen, Kapazitäten (Kondensatoren), Induktivitäten (tekercs) und Widerstände sind. Ein *Normalbaum* ist ein Spannbaum, der
alle Spannungsquellen,
keine Stromquellen,
möglichst viele Kapazitäten,
möglichst wenige Induktivitäten enthält.

Die Lösung ist eingeutig $\iff$ der Graph enthält einen Normalbaum.

Wie findet man einen Normalbaum?
Seien die Kosten der Kanten die folgenden:

$$k(\text{Spannungsquelle})=1;\ k(\text{Kapazität})=2;\ k(\text{Widerstand})=3;$$
$$k(\text{Induktivität})=4;\ k(\text{Stromquelle})=5$$

In dem gewichteten Graphen findet man einen minimalen Spannbaum mit z.B. dem Kruskal-Algorithmus.

Breitensuche oder BFS (Szélességi keresés)

Sei $G = (V, E)$ ein gerichteter/ungerichteter Graph, $x, y \in V$. y heisst *erreichbar* aus x, wenn es einen gerichteten/ungerichteten Weg von x nach y gibt.

Breitensuche

Input: gerichteter oder ungerichteter Graph G und Anfangsknoten $s \in V$.
Output: Menge der aus s erreichbaren Knoten.
Ablauf: Am Anfang ist s besucht, alle andere Knoten sind nicht besucht. Liste L enthält nur s.
Besuchen wir alle Knoten v für denen die Kante sv existiert und legen wir diese Knoten zu der Ende der Liste L und löschen wir s. s wird durchsucht.
In einem allgemeinen Schritt betrachten wir den ersten Knoten u auf Liste L. Besuchen wir alle nicht besuchte Knoten v für denen die Kante uv existiert und legen wir diese Knoten zu der Ende der Liste L und löschen wir u. u wird durchsucht.

Der Algorithmus terminiert wenn L leer ist. Das passiert wenn alle besuchte Knoten durchsucht werden.
Bemerkung. Falls in G alle Knoten von s aus erreichbar sind, betrachten wir für jeden Knoten außer s jene Kante, über die der Algorithmus zuerst den gegebenen Knoten erreicht. Dann bilden diese Kanten einen Spannbaum von G. Diesen nennen wir den *vom Algorithmus gelieferten Spannbaum* oder *BFS-Baum*.

Die Kanten des Bäumes werden als *Baumkanten (fa-élek)* bezeichnet.
Diejenigen Kanten, die nicht benutzt werden und von einem Knoten zu einem anderen Knoten im selben Teilbaum führen, der bei der Suche später besucht wird, heißen *Vorwärtskanten (előre-élek)*.
Diejenigen Kanten, die nicht benutzt werden und von einem Knoten zu einem anderen Knoten im selben Teilbaum führen, der bei der Suche bereits vorher besucht wurde, heißen *Rückwärtskanten (vissza-élek)*.
Diejenigen Kanten, die „quer“ von einem Teilbaum zu einem anderen Teilbaum verlaufen, heißen *Querkanten (kereszt-élek)*.

In ungerichteten Graphen gibt es nur Baumkanten, Querkanten und Vorwärtskanten.

Satz

(1) Der BFS-Baum enthält die kürzeste Wege von s zu alle Knoten die von s aus erreichbar sind. (2) Der Zeitbedarf des Algorithmus ist $konst \cdot (n + m)$.

Beweis.

(1) Vollständige Induktion nach der Länge des Weges; für 0 ist die Behauptung trivial.

Für Wege mit Länge k: zu dem vorletzte Knoten des Weges gibt es einen Weg im BFS-Baum mit Länge $k - 1$. Laut der Ablauf von BFS existiert dann einen Weg der Länge k zum Endknoten.

(2) Jeder von s aus erreichbare Knoten wird genau einmal besucht. Dabei werden alle aus ihm ausgehenden Kanten genau einmal untersucht. Die Anzahl der Schritte für einen Knoten v sind also $konst \cdot (1 + d(v))$.

Damit ist die Gesamtanzahl der Schritte $konst \cdot (n + \sum_{v \in V} d(v)) = konst \cdot (n + m)$.

Satz

Nach dem Ablauf von BFS gibt es keine Vorwärtskante.

Beweis. Folgt aus der Ablauf.

Anwendungen von BFS

1. Einen kürzesten gerichteten/ungerichteten Weg zwischen zwei beliebigen Knoten zu finden.
2. Entscheiden ob ein ungerichteter G zusammenhängend ist.
3. Die Komponente von einem ungerichteten G zu finden.